End - Term Report

Intelligent RRM

Inter IIT Tech Meet 14.0
Arista Networks – Problem Statement H4

Submitted by

Team 24

November 2025

Contents

1 System Evolution and Performance Outcomes

Modern RRM still depends mostly on AP-side observations collected using the same radio that handles client communication, leading to blind spots in client QoE, high airtime cost during periodic scans, and poor adaptability to diverse interference and device conditions. To overcome these limitations, the midterm of the problem statement targeted on developing an additional sensing radio per AP that performs continuous and intelligent spectrum scanning, prioritizing noisy channels while respecting DFS constraints. The system employs multi-armed bandits for scan scheduling, CNNs for non-Wi-Fi interference classification, and CUSUM/EWMA for noise-floor change detection, along with client-view acquisition and passive inference when 802.11k/v is unavailable. By combining AP-side and client-side insights, we steer transmit power, channel width, and OBSS-PD using a Bayesian optimisation framework to maximize throughput and QoE while maintaining network stability and guardrails.

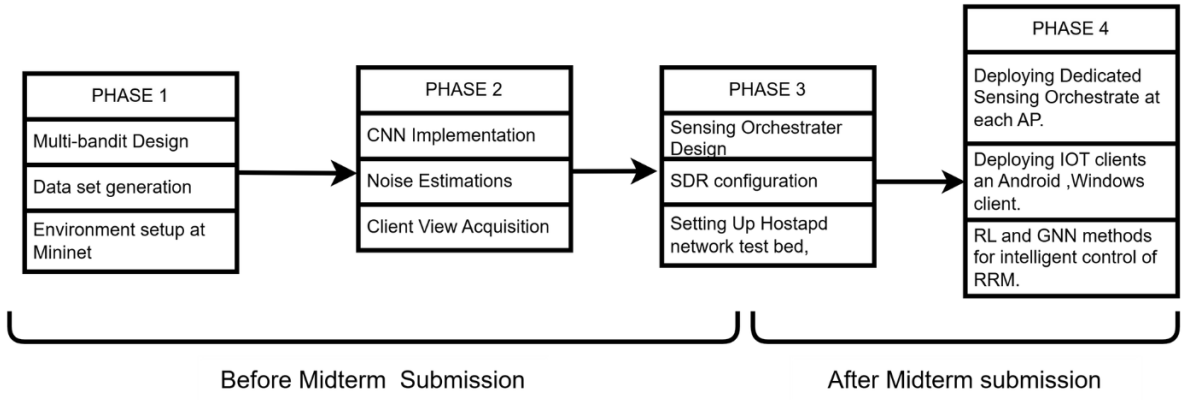


Figure 1: Problem statement Evolution

- In the first few phases, we built the core sensing stack in Mininet-WiFi for the network layer to evaluate BO-based optimization of QoE lift using client-view acquisition and Python-based testbed for validating multi-armed bandit design, CNN-based noise classifier.
- In the final phases we moved to a real testbed, designing a dedicated sensing orchestrator per AP with SDR-based spectrum capture and Hostapd based virtual APs.

So in the midterm both Network layer and physical layer were different due to the limitations, after developing an proof-of-concept we start deploying in real environment to integrate both together.

Limitations of Mininet-WiFi

- No real RF channel.
- Lack of real interference.
- Limited client diversity.
- Timing and scheduling artefacts.
- Protocol feature gaps.
- Unrealistic control-loop validation.

We had 3 Choices, but we leverage linux operating system Ubuntu 24.04 lts as AP.
We didn't use `Raspberry Pi 5` because the Wi-Fi chipset on the laptops supports Wi-Fi 6, whereas `Raspberry Pi 5` does not support this standard. Additionally, we couldn't use the institute's access points because we did not have API license access to the network controller.

2 Advantages of a Linux-based Virtual AP Testbed

- Real RF environment

- True client behaviour
- Full control of AP parameters
- Low-cost but realistic
- Better validation of safety and guardrails

3 Experimental Conditions and Setup Equipments

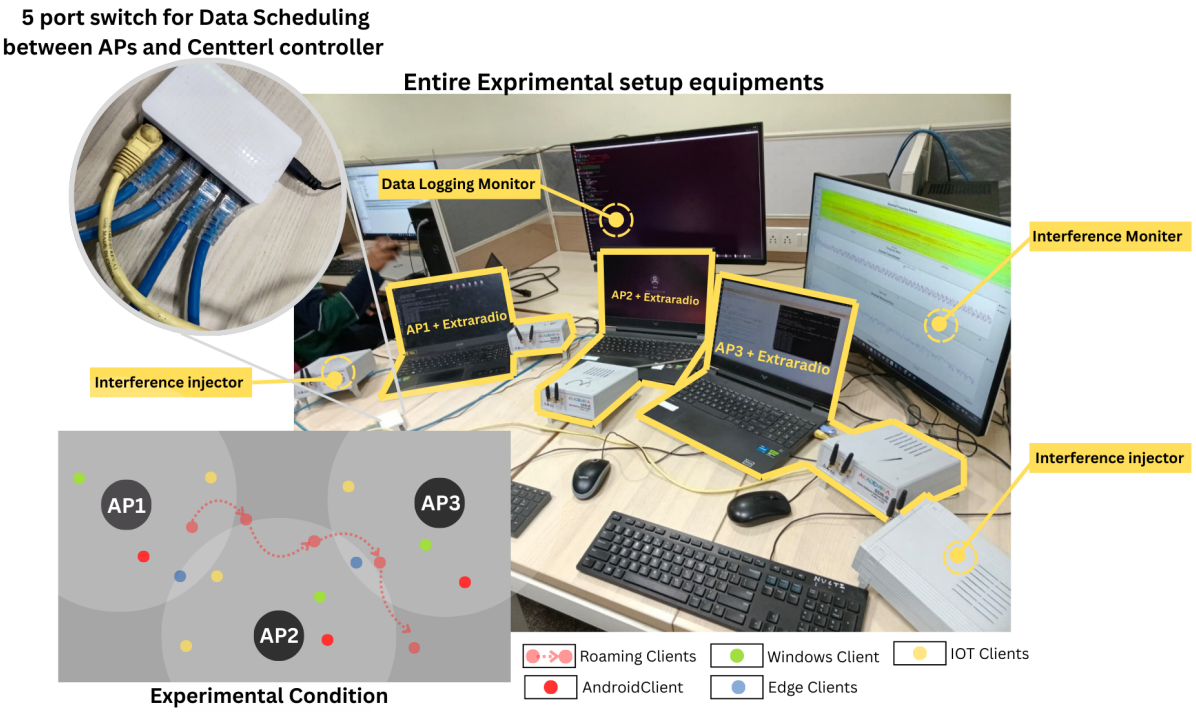


Figure 2: APs and GNU-RADIO Experimental setup

Equipment : 3 x HP Victus laptops, 5 x Akademica SDR B (3 as extra radio and 2 as Interference Generator) , 5-port ethernet switch (built-in CMCS CD protocol) for communication between APs and the central RL controller, 1 x HP Victus laptop for RL controller.

Environmental conditions: For training interference was generated as a probabilistic model around 2.44 GHz and 5.4 GHz central frequency, injecting different types of interference (Zigbee, BLE, Microwave, CW) into the environment with 10-second burst widths and a mean occurrence time of around 30 seconds. Next for 3 days log it was done in steps, first :

- Day 1: Low interference and stable environment, and low load per AP
- Day 2: Dense network with a mean period of occurrence of about 300s with 0.25 % probability of pure interference.
- Day 3: Sometimes a Dense network with some frequently roaming clients.

4 SDR-based Sensing Orchestrator

The entire Sensing Architecture for dynamic sensing and interference classifier was implemented in a real environment using Software Defined Radio (Akademika Pluto-SDR).

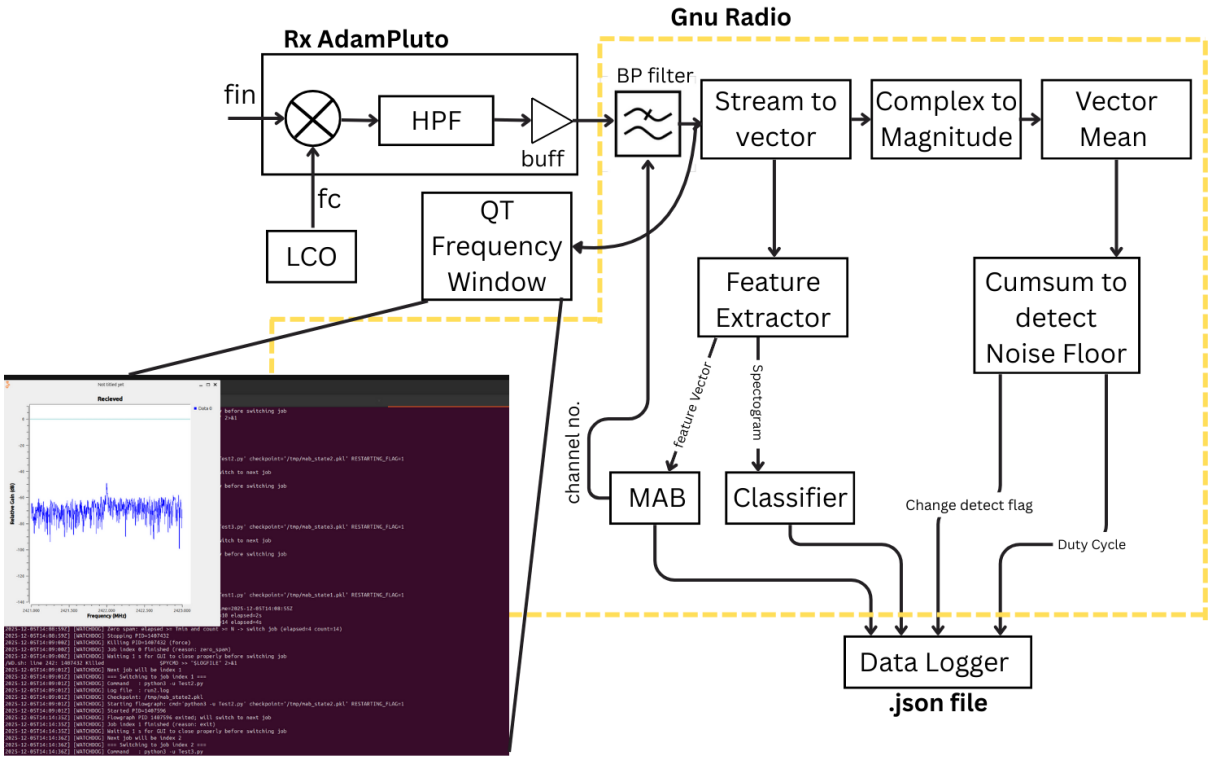


Figure 3: Additional sensing radio orchestrator using SDR and gnu radio.

- The GNU Radio pipeline performs bandpass filtering, vectorization, magnitude and mean computation and feature extraction from the raw PlutoSDR samples.
- These features feed the Multibandit (KalmanUCB) and classifier, while a CUSUM block [5] tracks the vector mean to detect noise-floor changes and generate change flags and duty-cycle hints.
- All decisions and measurements are logged to JSON, providing the input for our RL/GNN-based RRM controller and enabling quantitative evaluation of QoE/throughput gains.

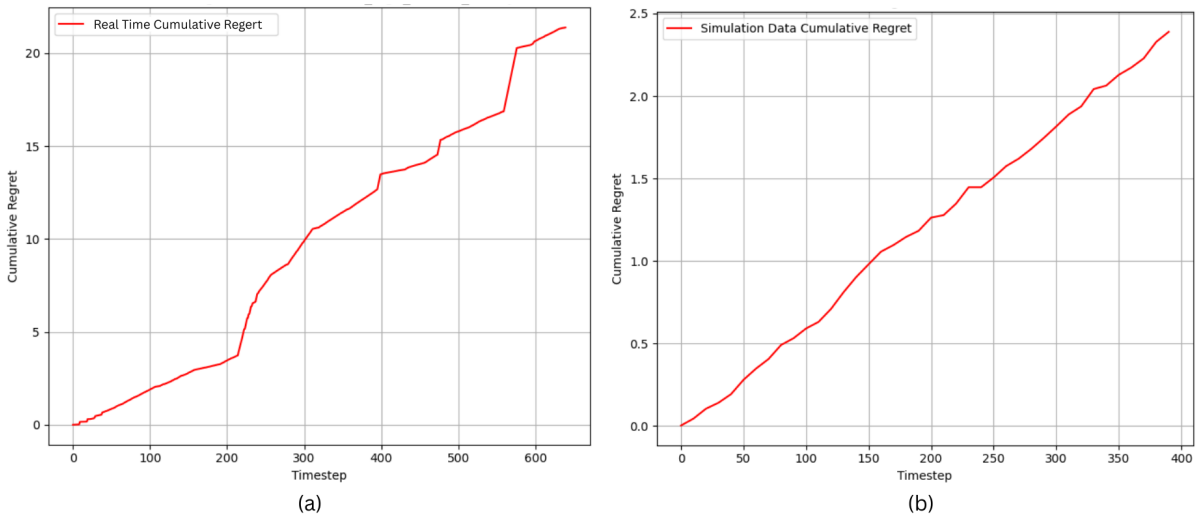


Figure 4: (a) Cumulative regret vs timesteps in real-time environment (b) Cumulative regret vs timesteps in simulation.

From the cumulative regret plots of both real-time simulation environments, it is confirmed that the real-time environment exhibits higher uncertainty and variability, leading to more abrupt increases in cumulative regret compared to the more stable simulation environment. Also, the Table of Band-wise Summary for both simulation and real-time environments is shown below. The average minimum time

sense of each channel is 50ms to 100ms, which reflects properly in the Real-Time sensing rather than the simulation environment, as temporal data are more accurate in actual environments and conditions are also genuine. Apart from that, DFS channels sense more time than the NON-DFS for effective radar detection.

Table 1: Simulation vs Real-Time Comparison Table

(a) Band-wise Summary for Real-Time Sensing

Channel State	Total	Avg Time	Regret/s	Reward/s
Non-DFS Channel	566	0.0300	1.1104	31.3547
NOP : DFS Channel	18	0.6111	0.0717	1.5239
AVAILABLE : DFS Channel	50	0.2000	0.1490	4.7265
RADAR : DFS Channel	6	0.1667	0.2223	5.6283

(b) Band-wise Summary for Simulation

Band	Total	Avg Time/ch	Regret/s	Reward/s
2.4GHz	832	0.075636	0.11890	0.02468
5GHz (Non-DFS)	1107	0.123000	0.16940	0.03270
5GHz (DFS)	61	0.003812	0.01280	0.00121

5 Multi-Timescale Control Loops for Wi-Fi Radio Resource Management

5.1 Fast Loop (Seconds–Minutes)

The fast loop operates as a real-time safety layer. It continuously ingests per-AP telemetry (e.g., DFS hits, interference levels, client load) and collapses it into a unified risk score for each candidate channel. On every tick, the controller automatically moves APs away from radar-hit or high-risk channels and steers clients toward cleaner, less-loaded spectrum, without requiring any manual intervention.

5.2 Slow loop (hours–days)

- **Role and timescale** Global RF planner running on an hours–days cadence (e.g., every $\geq$4 h), setting the long-term configuration for all APs.
- **Inputs**
  - Fused snapshot from the shared `Telemetry`: AP config (channel, BW, TX power, OBSS-PD), client stats (RSSI, retries, RTT bins), SDR channel summaries.
- **Core processing**
  - Enrich snapshot with RF-based AP–client distances and AP–AP "virtual RSSI" links.
  - Build interference/conflict graphs over APs [1].
  - Run DSATUR-based graph colouring to compute a global channel plan (spectral-overlap-aware, 2.4/5 GHz band-aware, and sticky to the previous plan).
  - Feed an RF-grounded interference graph into a GNN Q-network (CQL policy) to propose per-AP relative actions on TX power, bandwidth, and OBSS-PD [2].
- **Outputs**
  - MQTT actions: `set_channel`, `set_tx_power`, `set_bw`, `set_obss_pd` to update AP configs.
  - Logged experience tuples (snapshots, actions, rewards, explanations) for offline CQL training and analysis.

5.3 Event Loop:

Developing the event-loop controller had required working across the full stack of the system: configuring real APs, wiring up telemetry, replaying logs, and tuning guardrails until every exam and meeting

scenario behaved safely. We kept iterating on the controller logic, dashboards, and testbed runs until the actions were reproducible on hardware and easy to interpret from the operator's view.

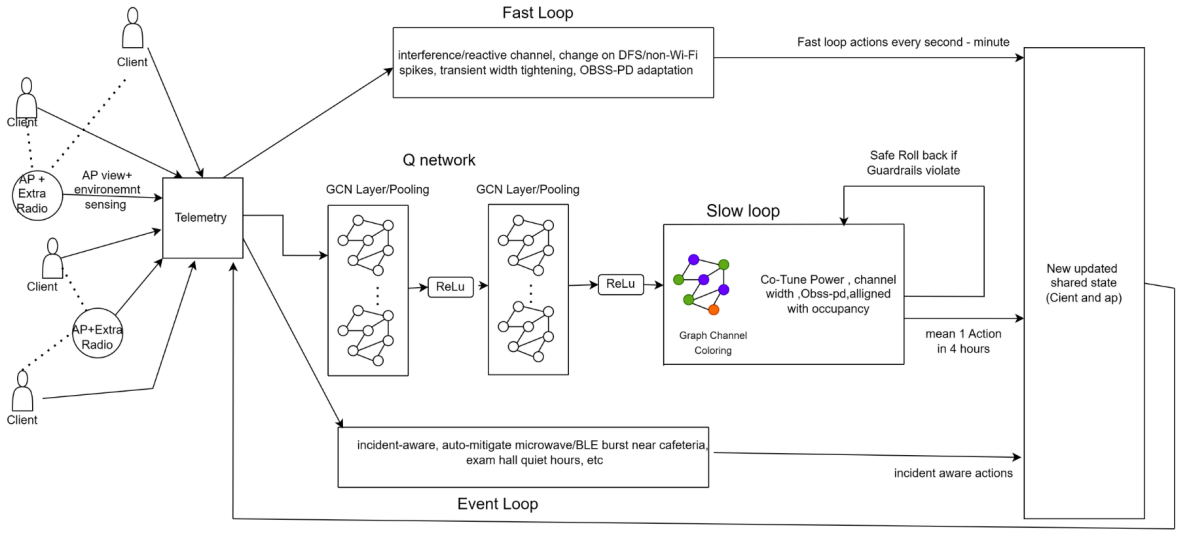


Figure 5: Multi timescale Control Loops.

Interference Graph Construction

3. Interference Graph Construction

The interference graph is built by calculating the RSSI values for both AP-client and AP-AP relations using the formulas above. The graph edges represent communication links, and the edge weights represent the RSSI, which is affected by both distance and interference.

6 Proof of Concept: Real-Time OBSS-PD Emulation

Motivation: While our Slow Loop operates on standard Linux SoftAPs, the Fast Loop OBSS-PD mechanism requires PHY-layer register access that is locked on standard consumer NICs (e.g., Intel AX210). To validate the decision-making logic without custom hardware, we devised a *Digital Twin* emulation strategy [3]. **Logic & "Weakest Client" Constraint:** We implemented a controller that constructs a real-time interference graph based on RF coupling rather than static topology. The algorithm calculates the maximum safe OBSS threshold (T_{OBSS}) using the constraint:

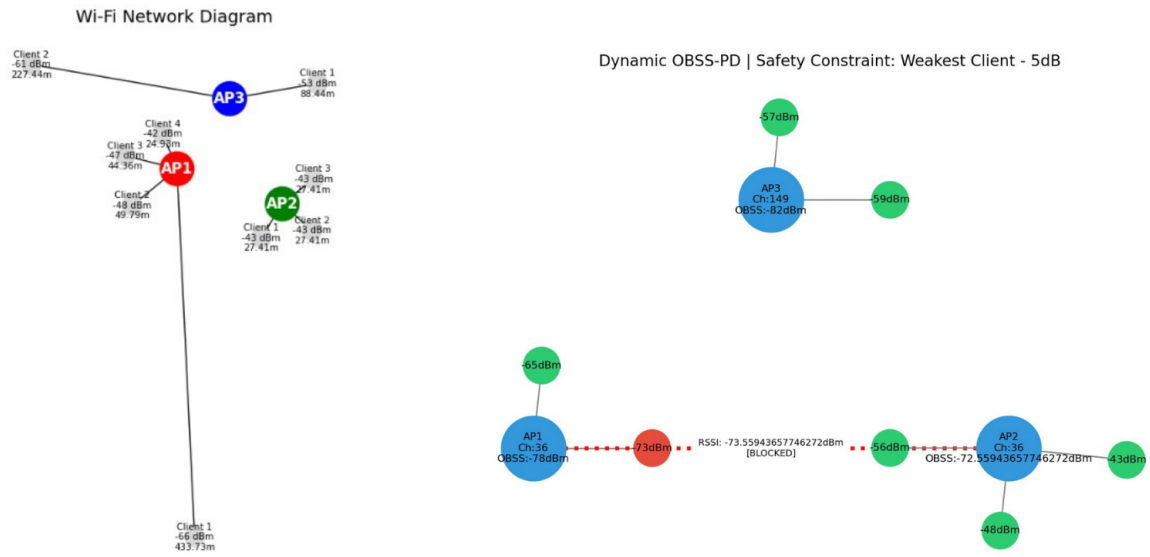
$$T_{OBSS} \leq RSSI_{weakest\_client} - \Delta safety\_margin \tag{1}$$

If the detected neighbor RSSI is below T_{OBSS}, the system flags the interaction as **SR ACTIVE** (Green), implying the neighbor can be treated as noise. If above, it is **BLOCKED** (Red), requiring standard backoff. **Emulation Mechanism:** We utilized **Linux Traffic Control (tc)** to proxy the physical layer impact. When the algorithm returns *SR ACTIVE*, artificial limits are removed, simulating full airtime utilization. When *BLOCKED*, netem injects latency to mimic the throughput penalty of CSMA/CA contention slots. This confirms that our Fast Loop correctly identifies spatial reuse opportunities in real-time.

7 Safe RL with Conservative Q-Learning (CQL)

The slow loop runs at an hours–days timescale and must satisfy three requirements:

- learn from *offline logs* (no unsafe online exploration initially),
- respect *hard safety constraints* (KPI regression bounds, change budgets, guardrails),



(a) (b)

Figure 6: **(a)** DSATUR graph coloring for channels. **(b)** Proof-of-Concept OBSS PD real-time interference graph.

- avoid overestimating Q-values for actions that are rarely or never observed in the logs.

This matches the setting of *Conservative Q-Learning* (CQL) [4]: an offline Q-learning algorithm that adds a penalty pushing Q-values of out-of-distribution actions downward, yielding a pessimistic and therefore safer policy on unseen actions.

MDP formulation. We model the slow loop as a Markov Decision Process (MDP) over slow-loop steps:

- **State** s_t: RF snapshot summarised as a graph $G_t = (V_t, E_t)$, with AP node features and AP–AP edge features derived from the interference graph.
- **Action** a_t: joint action across APs, implemented as per-AP discrete actions drawn from

  $$\mathcal{A} = \{\text{NO\_OP}, \text{POWER\_UP}, \text{POWER\_DOWN}, \text{BW\_UP}, \text{BW\_DOWN}, \text{OBSS\_UP}, \text{OBSS\_DOWN}\}.$$

- **Reward** r_t: global QoE reward (Section 7) derived from coverage, throughput, retries, fairness, and configuration churn.
- **Transition** s_{t+1}: new telemetry snapshot after the applied actions propagate through traffic dynamics and client mobility.

A GNN-based Q-network $Q_\theta(s, a)$ provides Q-values for each state–action pair.

Reward function

For each AP $i \in V_t$ at slow-loop step t we compute a per-AP QoE score, then aggregate across APs and add penalties. If one AP is starved, J drops and the global QoE is penalised, encouraging fair load distribution.

Final reward. The scalar reward at step t is

$$r_t = \text{effective\_QoE}_t - \lambda_{\text{churn}} \cdot \text{churn\_rate}_t - \lambda_{\text{retries}} \cdot \text{retries\_violation}_t, \tag{2}$$

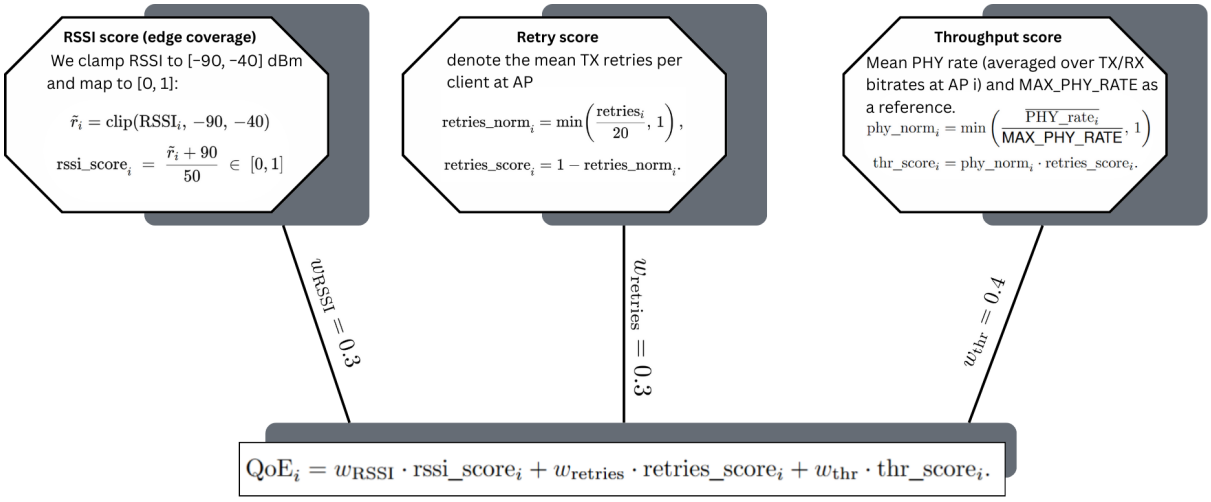


Figure 7: Defination of per AP QoE

with $\lambda_{\mathrm{churn}} = 0.3$ and $\lambda_{\mathrm{retries}} = 0.5$ in our implementation. This aligns with our intent: *maximize effective coverage and throughput with fairness, while minimizing retries and configuration churn.* In pretrain, we further subtract a fixed penalty when KPI guardrails are violated (e.g., edge throughput or latency regression), effectively shaping r_t to discourage unsafe regions of the configuration space [3].

CQL objective and justification

A typical CQL(H) objective for discrete actions is:

$$\begin{aligned}\min_{\theta} \quad & \alpha\Big(E_{s\sim\mathcal{D}}\big[\log \sum_{a'} \exp Q_\theta(s,a')\big] - E_{(s,a)\sim\mathcal{D}}\big[Q_\theta(s,a)\big]\Big) \\ & + E_{(s,a,r,s')\sim\mathcal{D}}\big[\big(Q_\theta(s,a) - y\big)^2\big],\end{aligned} \tag{3}$$

where $\alpha > 0$ controls the strength of conservatism.

- The first term $E_s\big[\log \sum_{a'} \exp Q_\theta(s,a')\big] - E_{(s,a)}[Q_\theta(s,a)]$ penalises Q-functions that assign large values to actions that are not supported by the dataset.
- The second term is the standard TD error, which keeps Q-values accurate on the behaviour policy.

In our RRM deployment:

- We trained our CQL policy by collecting offline dataset as multiple hours episodes from slow and fast loop that ensures safe actions implemented with guardrails.
- CQL naturally prefers actions that are well-supported by these logs and penalises aggressive, unseen combinations of channel, power, bandwidth and OBSS-PD.
- This matches the operational risk profile of a live Wi-Fi network: we seek small, reliable improvements over the baseline, not unconstrained exploration.

8 Causal Inference: Uplift Modeling for QoE Impact

For each action (e.g., `POWER_DOWN`, `OBSS_UP`), the controller logs AP features and pre-change KPIs.

9 Explainability: per-change reason codes.

Every slow-loop decision is accompanied by a human-readable explanation generated from the same features used by the GNN and guardrails. For each AP, the controller records the chosen discrete

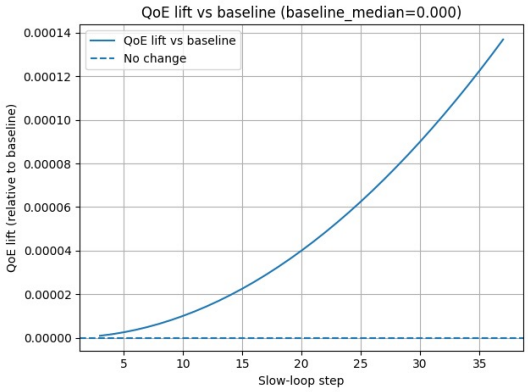


Figure 8: Day1 qoe

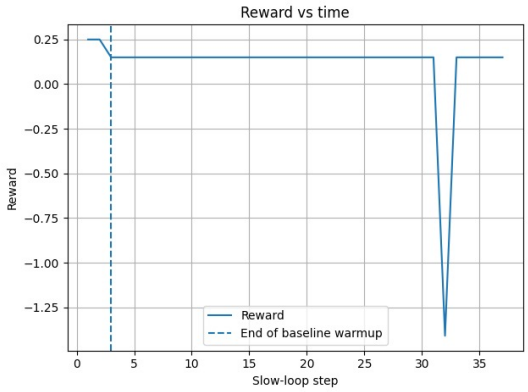


Figure 9: Day1 Reward

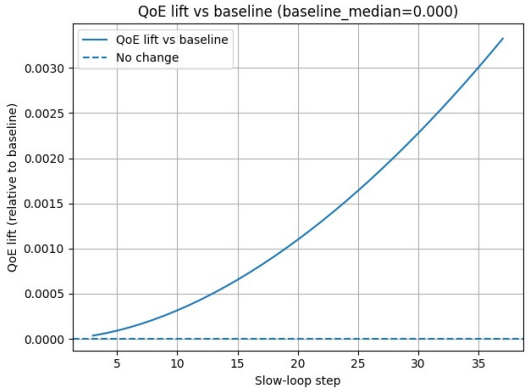


Figure 10: Day2 qoe

Figure 11: Day2 Reward

action, the local metrics (client count, minimum RSSI, mean retries, summed interference weight), and the Q-values for all legal actions. These are turned into structured *reason codes*, for example:

Example reason codes. During deployment, the controller emits per-AP explanations in a compact machine- and human-readable format. For example, in one slow-loop step we observed:

10 Operational Readiness

To ensure the system is suitable for production deployment, we integrated comprehensive mechanisms for Service Level Objective (SLO) compliance, privacy, stability management, and the constraints/restrictions we had in implementing such hard acceptance criteria and guardrails.

10.1 SLO Breach Management and Audit

To maintain network stability, automatic rollback mechanisms are crucial when SLO breaches occur (e.g., detected performance degradation). All automated decisions and configuration changes are logged to an immutable audit trail. This enables a rapid diagnosis and allows the controller to quickly revert to a previous stable state if the KPIs deteriorate, minimizing the duration of any negative client impact.

10.2 Privacy and Data Retention

Client privacy is enforced at the data ingestion layer. All MAC addresses are hashed to anonymize user identities, with support for opt-out mechanisms. Furthermore, strict data retention policies are applied to limit the storage duration of fine-grained telemetry, ensuring compliance with data protection standards.

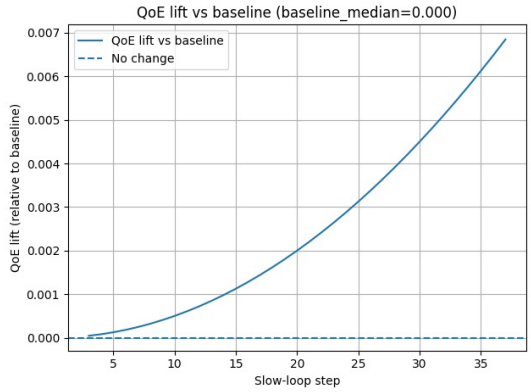


Figure 12: Day3 qoe

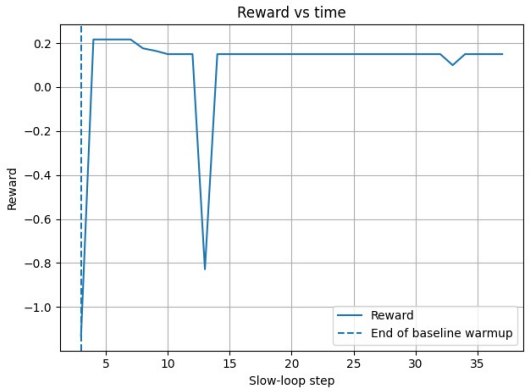


Figure 13: Day3 Reward

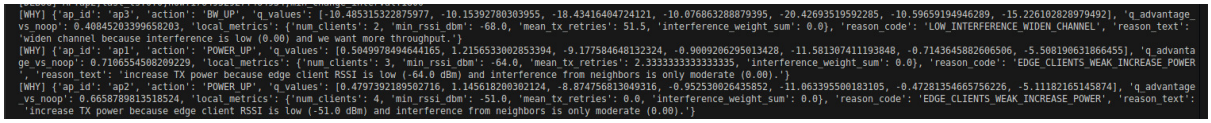


Figure 14: Explainability report.

10.3 Balancing QoE Gains vs. Stability

We tried to enforce strict guardrails to ensure that QoE improvements do not compromise system stability. To prevent oscillation, the controller adheres to a "configuration churn" budget . If a change results in a breach of safety thresholds—such as a drop in edge-client throughput—the system automatically triggers a rollback to prioritize connectivity over optimization.

10.4 Constraints

For fast roaming, we need the support of 802.11r on client devices.In addition to that, we need 802.11k,v support on both the AP and the client device. We used 3 linux laptops as APs whose chipset mentions support of 802.11k and v but we could not get this support from the driver mt76. Hence,we could not use the benefit of 802.11k,v in fast roaming. RU utilization and OFDMA scheduling were not done because it is a feature of Wifi 6 and requires advanced driver support, which our driver mt76 lacked.

11 Advanced Client View

The deployment hardware did not support IEEE 802.11mc (FTM), so fine-timing–based RTT / location measurements were not available. To overcome this limitation, we implemented a passive, transport-layer client-view framework built on TCP timestamp options. For each client flow, packets were mirrored and RTT was computed by correlating outgoing TCP timestamp values (TSval) with the echoed timestamps (TSecr) in the corresponding ACKs. When timestamp options were missing or inconsistent, we fell back to capture-time differences between request/response pairs. For every client, we then derived:

- median RTT,
- P95 RTT,
- loss rate and loss variance.

To recover coarse spatial information in the absence of 802.11mc, we bucketed clients by RSSI into three bins:

- **Near:** RSSI $> -45\,\mathrm{dBm}$,
- **Mid:** $-65\,\mathrm{dBm} \leq$ RSSI $\leq -45\,\mathrm{dBm}$,
- **Edge:** $-75\,\mathrm{dBm} \leq$ RSSI $< -65\,\mathrm{dBm}$.

These RTT, loss and RSSI-bin metrics together provide a comprehensive client-side QoE profile even without 802.11mc support. In addition, we applied a simple CUSUM-based spike detector on the RTT time series to flag sudden latency increases. This end-to-end pipeline allowed us to:

- map RTT behaviour to spatial RSSI bins,
- detect interference or coverage problems,
- and assess the stability of the client wireless experience over time.

Figure 15: Advance Client View workflow.

References

[1] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[2] M. Eisen and A. Ribeiro, "Optimal Wireless Resource Allocation With Random Edge Graph Neural Networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2977–2991, 2020.

[3] B. Bellalta, "IEEE 802.11ax: High-Efficiency WLANs," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 38–46, 2016.

[4] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-Learning for Offline Reinforcement Learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1179–1191, 2020.

[5] E. S. Page, "Continuous Inspection Schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[6] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge University Press, 2020.

[7] IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High Efficiency WLAN, *IEEE Std 802.11ax-2021*, 2021.

A Acronyms and Abbreviations

AFC Automated Frequency Coordination

AP Access Point

BLE Bluetooth Low Energy

BO Bayesian Optimization

BSS Basic Service Set

BSS-TM BSS Transition Management (802.11v)

CCA Clear Channel Assessment

CNN Convolutional Neural Network

CQL Conservative Q-Learning

CUSUM Cumulative Sum (Change Detection Algorithm)

DFS Dynamic Frequency Selection

EIRP Equivalent Isotropically Radiated Power

EWMA Exponentially Weighted Moving Average

FHSS Frequency Hopping Spread Spectrum

FT Fast Transition (802.11r)

FTM Fine Timing Measurement (802.11mc)

GNN Graph Neural Network

IoT Internet of Things

KPI Key Performance Indicator

MAC Media Access Control

MCS Modulation and Coding Scheme

MDM Mobile Device Management

MPDU MAC Protocol Data Unit

MU-MIMO Multi-User Multiple Input Multiple Output

OBSS-PD Overlapping Basic Service Set - Preamble Detection

OFDMA Orthogonal Frequency-Division Multiple Access

OUI Organizationally Unique Identifier

PER Packet Error Rate

PHY Physical Layer

PII Personally Identifiable Information

PSC Preferred Scanning Channel (6 GHz)

QoE Quality of Experience

QoS Quality of Service

QUIC Quick UDP Internet Connections

RL Reinforcement Learning

RRM Radio Resource Management

RSSI Received Signal Strength Indicator

RTT Round Trip Time

RU Resource Unit

SINR Signal-to-Interference-plus-Noise Ratio

SLA Service Level Agreement

SLO Service Level Objective

SNR Signal-to-Noise Ratio

SR Spatial Reuse

SSID Service Set Identifier

TCP Transmission Control Protocol

TPC Transmit Power Control